Installing OpenShift through Foreman

All you need is CoreOS

Jan Bundesmann

February 6th, 2023

THE Linux & Open Source Company!

Consulting

Engineering

Support

Training

AlmaLinux, amazon web services, ANSIBLE, Azure, CentOS, debian, docker, FOREMAN, git, Google Cloud Platform

Istio, kafka, KATELLO, kubernetes, KVM, ORACLE Linux, oVirt, PROXMOX, puppet, RANCHER

Red Hat, RED HAT OPENSHIFT, Rocky Linux, SALTSTACK, SUSE, Terraform, Ubuntu, vmware, Windows

What?

- Trainer for OpenShift
- Infrastructure for participants on OpenShift 4.X
- Refined approach at deploying CoreOS through Foreman / orcharhino
- Packaged templates seem outdated (CoreOS CoreOS not RedHat CoreOS)

Agenda

1 CoreOS
- What is CoreOS?
- Provisioning of CoreOS
- Prepare foreman for CoreOS provisioning

2 OpenShift and CoreOS

3 Conclusions and Outlook

Agenda

1 CoreOS
- What is CoreOS?
- Provisioning of CoreOS
- Prepare foreman for CoreOS provisioning

2 OpenShift and CoreOS

3 Conclusions and Outlook

What is CoreOS?

- Immutable minimal version of Fedora Linux / Red Hat Enterprise Linux
- Automatic upgrades through `rpm-ostree` and `zincati`
- Ignition for adjustments during installation
  - Environment for unattended installation
  - Partitioning
  - Creates files (systemd, network config, …)

Demo

Upgrades / Rollbacks

Let me show you some magic

Resembles git, huh?

Agenda

1 CoreOS
- What is CoreOS?
- Provisioning of CoreOS
- Prepare foreman for CoreOS provisioning

2 OpenShift and CoreOS

3 Conclusions and Outlook

Installation principle

1. Boot installer medium
   - live medium
   - vm / cloud image
   - PXELinux with Kernel and initrd
2. start `coreos-installer`
3. copy filesystem image to hard disk
4. reboot
5. launch ignition for customizations

Time: < 3 minutes

Installation from live medium

- Fedora CoreOS available at getfedora.org/en/coreos
- RHCOS available at mirror.openshift.com/pub/openshift-v4/dependencies/rhcos

Note: RHCOS is only available with a RedHat subscription.

1. Boot machine from live iso
2. Run `coreos-installer install /dev/sda --ignition-url https://some.thing/ignition.ign`
3. Reboot
4. Enjoy!

Demo time

Just if time permits and you're interested

What are ignition files

- JSON files
- Most cases: Ignition specification 3.2.0
- Equivalent to Kickstart / Preseed / Autoyast / Autoinstall
- My first impression: https://xkcd.com/927

For live / individual installation:

- Usually not written manually, created from `butane`.
- `butane` transform YAML input to ignition (JSON).
- `butane`-input has to obey specifications for target (Fedora CoreOS, OpenShift 4.x, FlatCar Linux)

Demo time

How to use butane

Provision CoreOS via network boot

- We require `coreos.inst.<something>` to trigger running the installer.
- Some available parameters (→ full list):
  - `coreos.inst.install_dev` - where to install
  - `coreos.inst.stream` - online installation; next, testing, stable
  - `coreos.inst.image_url` - offline installation; specify image
  - `coreos.inst.ignition_url` - where is ignition
  - `coreos.inst.skip_reboot` - Don't reboot after installing
- We need **kernel**, **initrd**, **rootfs** and **ignition file**.

Practical approach

```
url=$(curl --silent \
  https://builds.coreos.fedoraproject.org/streams/stable.json | \
  jq -r .architectures.x86_64.artifacts.metal.formant.iso.disk.location)

curl ${url} -o coreos.iso
```

- Boot files under `images/pxeboot` (as with normal RHEL installation media)
- Kernel and initrd image can be published via TFTP.
- Publish rootfs via HTTP - we need an HTTP server for the ignition files anyways.

This approach also works on Foreman.

Agenda

1 CoreOS
- What is CoreOS?
- Provisioning of CoreOS
- Prepare foreman for CoreOS provisioning

2 OpenShift and CoreOS

3 Conclusions and Outlook

Actions in foreman

- **Installation medium**: pointing to CoreOS iso folder, type RedHat
- **PXELinux template** containing `coreos.inst.` parameters
- **Provisioning template** for ignition file
- Foreman expects a partition table for each OS, so:
  - Fake **partition table**
  - This is already present.
  - We could write a real one, but that's for later.
- Create **operating system** for CoreOS and assign templates as well as installation medium
- (Optional) create **hostgroups**

ansible code: create operating system

We start with the operating system so we can reference it later:

```yaml
- name: "Create Operating System CoreOS"
  theforeman.foreman.operatingsystem: &os
    <<: *connection
    name: CoreOS
    major: "4"
    minor: "11"
    os_family: Redhat
    state: present
    password_hash: MD5
    architectures:
      - x86_64
    ptables:
      - CoreOS default fake
```

Installation medium

- Download ISO
- Mount it to a well-defined place
- Add directory in apache configs

ansible code: installation medium

```yaml
- name: create directory for installation media
  file:
    path: /var/www/html/pub/installation_media/coreos/iso
    state: directory
- name: download coreos live image
  command:
    cmd: wget https://some.fqdn/latest/coreos.iso
    chdir: /var/www/html/pub/installation_media/coreos
    creates: /var/www/html/pub/installation_media/coreos/live.iso
- name: mount coreos live image
  command:
    cmd: mount -o loop live.iso iso
    chdir: /var/www/html/pub/installation_media/coreos
    creates: /var/www/html/pub/installation_media/coreos/iso/isolinux

- name: create installation medium in foreman
  theforeman.foreman.installation_medium:
    <<: *connection
    name: coreos
    operatingsystems:
      - CoreOS
    os_family: Redhat
    path: http://your.foreman.fqdn/pub/installation_media/coreos/iso

- name: "Create Operating System CoreOS"
  theforeman.foreman.operatingsystem:
    <<: *os
    media:
      - coreos
```


www.atix.de

Template for Ignition file I

```
<%#
kind: provision
name: Ignition File for CoreOS
model: ProvisioningTemplate
oses:
- CoreOS
%><%
  method = 'POST'
  url = foreman_url('built')
  curl_opts = ["-H 'Content-Type: text/plain'"]
  if url.start_with?('https')
    curl_opts << "--insecure"
  end

  raw_script_command = "/usr/bin/curl " + curl_opts.join(' ') + " --silent " + url

  merge_pre = '"merge": [{"compression": "gzip", "source": "data:; base64,'
  merge_post = '"}]'
  if (host_param('additional_ignition_packed') && host_param('additional_ignition_packed') != "")
    additional_ignition = merge_pre + host_param('additional_ignition_packed') + merge_post
  else
    additional_ignition = ""
  end
-%>
```

Template for Ignition file II

```
{
  "ignition": {
    "config": {
      <%= additional_ignition %>
    },
    "version": "3.1.0"
  },
  "passwd": {
    "users": [
      {
        "name": "core",
        "sshAuthorizedKeys": [
          "<%= host_param('remote_execution_ssh_keys').first %>"
        ]
      }
    ]
  },
  "systemd": {
    "units": [
      {
        "contents": "[Unit]\nDescription=Tell orcharhino I'm ready\nAfter=network-online.target\n\n[Service]\
            nType=simple\nExecStart=<%= raw_script_command %>\nRestartSec=5\nRestart=on-failure\n\n[Install]\
            nWantedBy = multi-user.target\n",
        "enabled": true,
        "name": "built.service"
      }
    ]
  }
}
```

Caveats

- Normal scenario
  - network boot by default
  - as soon as host is ‘built’, chainload boot loader on local disk
- CoreOS:
  - no method to call command during first stage
  - call home only implemented as SystemD Unit for first boot
- Workaround
  - boot from local drive by default
  - only use network boot if no os is present (and host is in build mode)
- Errors in ignition file are not prominently displayed.
  - You can add additional ignition files as host parameters.

PXELinux template

```
<%#
kind: PXELinux
name: CoreOS PXELinux for OpenShift
oses:
- CoreOS
%>
DEFAULT local_chain_hd0

TIMEOUT 20
PROMPT 0
ONERROR coreos
ONTIMEOUT coreos

LABEL local_chain_hd0
  MENU LABEL Chainload the first hard drive (hd0)
  COM32 chain.c32
  APPEND hd0

LABEL coreos
    KERNEL <%= @kernel %>
    APPEND initrd=<%= @initrd %> <%
     %>coreos.live.rootfs_url=http://<%= foreman_server_fqdn() %>/pub/installation_media/coreos/iso/images/
          pxeboot/rootfs.img <%
     %>nomodeset rd.neednet=1 coreos.inst.install_dev=/dev/vda <%
     %>coreos.inst.ignition_url=<%= foreman_url('provision') %>
IPAPPEND 2
```

ansible code: create templates

```yaml
- name: Copy templates to target host
  copy:
    src: "{{ item }}"
    dest: /tmp
  loop:
    - ignition.json.erb
    - pxelinux.erb

- name: Create template from ignition file
  theforeman.foreman.provisioning_template:
    <<: *connection
    template: '{{ lookup("file", "files/ignition.json.erb") }}'
    operatingsystems:
      - CoreOS

- name: Create PXELinux template
  theforeman.foreman.provisioning_template:
    <<: *connection
    template: '{{ lookup("file", "files/pxelinux.erb") }}'
    operatingsystems:
      - CoreOS
```

ansible code: finish configuration of operating system

```yaml
- name: "Associate ignition file with CoreOS"
  theforeman.foreman.os_default_template:
    <<: *orcharhino_connection
    operatingsystem: "CoreOS"
    template_kind: "provision"
    provisioning_template: "Ignition File for CoreOS"

- name: "Associate ignition file with CoreOS"
  theforeman.foreman.os_default_template:
    <<: *orcharhino_connection
    operatingsystem: "CoreOS"
    template_kind: "PXELinux"
    provisioning_template: "CoreOS PXELinux for OpenShift"
```

Agenda

1 CoreOS
- What is CoreOS?
- Provisioning of CoreOS
- Prepare foreman for CoreOS provisioning

2 OpenShift and CoreOS

3 Conclusions and Outlook

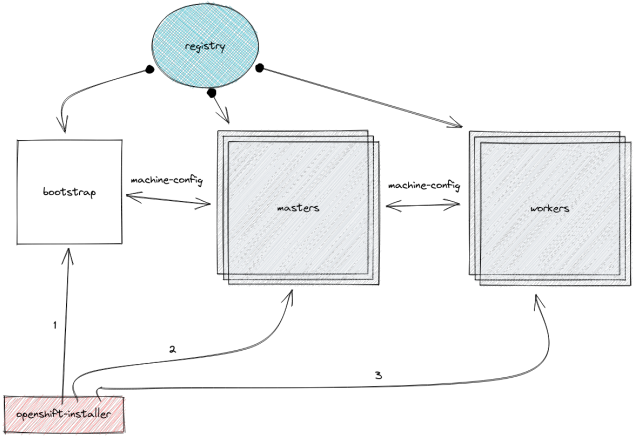
What is OpenShift?

- Kubernetes distribution
- DevOps platform leveraging Kubernetes
- Downstream version of OKD

Why CoreOS

- Holistic approach
- Container applications use an immutable approach
- Container platforms are immutable
- Let container infrastructure be immutable as well

Installation scheme



Installation variants

Automatic installation

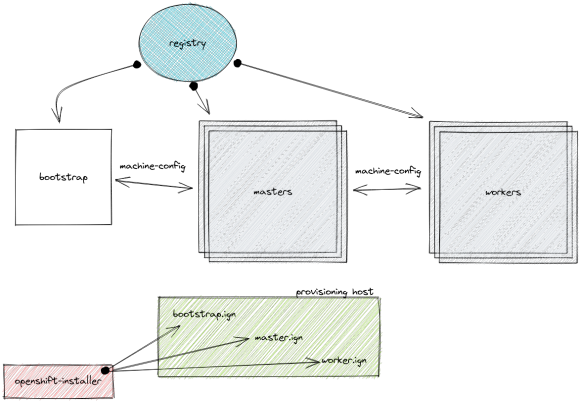
- AWS, GCE, Azure
- vSphere, RHV
- OpenStack

User-provisioned infrastructure

- Bare-metal
- Any hypervisor

Let `Foreman` help you!

User-provisioned infrastructure



hosts

1. bootstrap node (temporary)
2. master nodes
3. worker nodes

network / load balancer

- api.cluster.domain: masters
- api-int.cluster.domain: masters
- *.apps.cluster.domain: workers

deployment automatism for CoreOS

Set DNS entries

```yaml
- name: obtain key_secret for nsupdate
  ansible.builtin.shell: awk -F '"' '/secret/{print $2}' /etc/bind/rndc.key
  register: rndc_key
  changed_when: false

- name: perform dns settings
  community.general.nsupdate:
    key_algorithm: "hmac-sha256"
    key_secret: "{{ rndc_key.stdout }}"
    key_name: rndc-key
    server: 'localhost'
    record: "{{ item.record }}"
    zone: <DOMAIN>
    type: "{{ item.type | default(omit) }}"
    ttl: "{{ item.ttl | default(86400) }}"
    value: <LOAD-BALANCER>
  loop:
    - record: 'apps'
    - record: 'api'
    - record: 'api-int'
    - record: '*.apps'
      type: 'CNAME'
      value: 'apps'
```


www.atix.de

Ansible code: the openshift installer

```yaml
- name: "Create folder structure for openshift installation"
  file:
    path: /opt/openshift-installer
    state: directory

- name: Download and unpack openshift client
  ansible.builtin.unarchive:
    src: "https://path.to/openshift-client-linux.tar.gz"
    dest: /usr/local/bin
    creates: /usr/local/bin/oc
    remote_src: yes

- name: "Download OpenShift installer to recently created folder"
  ansible.builtin.get_url:
    url: "https://path.to/openshift-install-linux.tar.gz"
    dest: /opt/openshift-installer/openshift-install-linux.tar.gz

- name: "unpack installer"
  ansible.builtin.unarchive:
    src: /opt/openshift-installer/openshift-install-linux.tar.gz
    dest: /opt/openshift-installer/
    remote_src: yes
```

Create a config file

The main input for the openshift installer:

```yaml
---
apiVersion: v1
baseDomain: atix-training.de
metadata:
  name: openshift
compute:
  - hyperthreading: Enabled
    name: worker
    replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  none: {}
pullSecret: '{{ pullSecret | from_yaml | to_json }}'
```


www.atix.de

Ansible code: create config file

```yaml
- name: "Delete config folder"
  file:
    path: /opt/openshift-installer/cluster
    state: absent

- name: "Recreate config folder"
  file:
    path: /opt/openshift-installer/cluster
    state: directory

- name: "create config file"
  vars:
    pullSecret: "{{ lookup('file', 'PULL_SECRET') }}"
  template:
    src: config.yaml
    dest: /opt/openshift-installer/cluster/install-config.yaml
```

Create manifests

```
./openshift-install create manifests --dir=cluster
```

Result
- no more config file
- folder full of yaml manifests for control plane and machine config

Note This will consume the config file

Modify manifests

For instance turn controle plane non schedulable

```
sed -i 's/mastersSchedulable: true/mastersSchedulable: false' \
      ./cluster/manifests/cluster-scheduler-02-config.yml
```

Ansible code: create and modify manfests

```yaml
- name: "create manifests"
  command:
    cmd: ./openshift-install create manifests --dir=cluster
    chdir: /opt/openshift-installer/
    creates: /opt/openshift-installer/cluster/manifests/cluster-config.yaml

- name: "modify manifests: turn master non-schedulable"
  lineinfile:
    line: "  mastersSchedulable: false"
    path: /opt/openshift-installer/cluster/manifests/cluster-scheduler-02-config.yml
    regexp: "  mastersSchedulable: true"
```

Create ignition files

```
./openshift-install create ignition-configs --dir=cluster
```

Result
- three ignition files
- manifest files are gone

Note Template expects base64 encoded version of additional gzipped ignition.

Ansible code: create ignition files

```
- name: "create ignition files"
  command:
    cmd: ./openshift-install create ignition-configs --dir=cluster
    chdir: /opt/openshift-installer/
    creates: /opt/openshift-installer/cluster/bootstrap.ign
```

And now something funny

Direct approach

- `cat bootstrap.ign |gzip |base64 -w 0`
- `gzip -c bootstrap.ign |base64 -w 0`

- Output differs
- Inclusion using butane still different
- Installation fails without notification

Ansible code: create additional ignition

```yaml
- name: "create additional ignition configs for actual deployment"
  shell:
    cmd: >
      podman run -v ./:/data --interactive --rm
      quay.io/coreos/butane:latest
      --files-dir /data/ -r --pretty --strict
      < butane_{{ role }}.yaml
      | jq -r .ignition.config.merge[].source | cut -d , -f 2
    chdir: /opt/openshift-installer/cluster
  loop:
    - bootstrap
    - master
    - worker
  loop_control:
    loop_var: role
  register: butane_output

- name: "set facts"
  set_fact:
    bootstrap_ignition: "{{ butane_output.results[0].stdout }}"
    master_ignition: "{{ butane_output.results[1].stdout }}"
    worker_ignition: "{{ butane_output.results[2].stdout }}"
```


www.atix.de

Ansible code: create bootstrap node

```yaml
- name: "create bootstrap node"
  theforeman.foreman.host:
    <<: *orcharhino_connection
    name: "bootstrap.openshift.atix-training.de"
    hostgroup: ocp_controlplane
    parameters:
      - name: additional_ignition_packed
        parameter_type: string
        value: "{{ bootstrap_ignition }}"
    organization: "{{ or_organization }}"
    location: "{{ or_location }}"
    build: yes

- name: "start bootstrap node"
  theforeman.foreman.host_power:
    <<: *orcharhino_connection
    name: "bootstrap.openshift.atix-training.de"
    state: on

- name: "wait for bootstrap to complete and API become available"
  wait_for:
    host: "api-int.openshift.atix-training.de"
    port: 6443
    delay: 10
```

Agenda

1 CoreOS
- What is CoreOS?
- Provisioning of CoreOS
- Prepare foreman for CoreOS provisioning

2 OpenShift and CoreOS

3 Conclusions and Outlook

Summary

- Some major problems need solution before PR
- CoreOS can be handled by Foreman, still difficult
- OpenShift is installed easily, once CoreOS is set up

https://atix.de/config-management-camp-2023/